

UNITED STATES UTILITY PATENT APPLICATION

**AN ALGORITHM-TO-HARDWARE SYSTEM AND METHOD FOR CREATING A
DIGITAL CIRCUIT**

INVENTOR:

Richard J. CARTER
276 Angela Drive
Los Altos, CA 94022

AN ALGORITHM-TO-HARDWARE SYSTEM AND METHOD FOR CREATING A
DIGITAL CIRCUIT

5 FIELD OF THE INVENTION

This invention relates to creating a digital circuit. More particularly, the invention relates to converting an algorithm rendered in a programming language into an actual hardware implementation consisting of the digital circuit.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995
1000
1005
1010
1015
1020
1025
1030
1035
1040
1045
1050
1055
1060
1065
1070
1075
1080
1085
1090
1095
1100
1105
1110
1115
1120
1125
1130
1135
1140
1145
1150
1155
1160
1165
1170
1175
1180
1185
1190
1195
1200
1205
1210
1215
1220
1225
1230
1235
1240
1245
1250
1255
1260
1265
1270
1275
1280
1285
1290
1295
1300
1305
1310
1315
1320
1325
1330
1335
1340
1345
1350
1355
1360
1365
1370
1375
1380
1385
1390
1395
1400
1405
1410
1415
1420
1425
1430
1435
1440
1445
1450
1455
1460
1465
1470
1475
1480
1485
1490
1495
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1555
1560
1565
1570
1575
1580
1585
1590
1595
1600
1605
1610
1615
1620
1625
1630
1635
1640
1645
1650
1655
1660
1665
1670
1675
1680
1685
1690
1695
1700
1705
1710
1715
1720
1725
1730
1735
1740
1745
1750
1755
1760
1765
1770
1775
1780
1785
1790
1795
1800
1805
1810
1815
1820
1825
1830
1835
1840
1845
1850
1855
1860
1865
1870
1875
1880
1885
1890
1895
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
2055
2060
2065
2070
2075
2080
2085
2090
2095
2100
2105
2110
2115
2120
2125
2130
2135
2140
2145
2150
2155
2160
2165
2170
2175
2180
2185
2190
2195
2200
2205
2210
2215
2220
2225
2230
2235
2240
2245
2250
2255
2260
2265
2270
2275
2280
2285
2290
2295
2300
2305
2310
2315
2320
2325
2330
2335
2340
2345
2350
2355
2360
2365
2370
2375
2380
2385
2390
2395
2400
2405
2410
2415
2420
2425
2430
2435
2440
2445
2450
2455
2460
2465
2470
2475
2480
2485
2490
2495
2500
2505
2510
2515
2520
2525
2530
2535
2540
2545
2550
2555
2560
2565
2570
2575
2580
2585
2590
2595
2600
2605
2610
2615
2620
2625
2630
2635
2640
2645
2650
2655
2660
2665
2670
2675
2680
2685
2690
2695
2700
2705
2710
2715
2720
2725
2730
2735
2740
2745
2750
2755
2760
2765
2770
2775
2780
2785
2790
2795
2800
2805
2810
2815
2820
2825
2830
2835
2840
2845
2850
2855
2860
2865
2870
2875
2880
2885
2890
2895
2900
2905
2910
2915
2920
2925
2930
2935
2940
2945
2950
2955
2960
2965
2970
2975
2980
2985
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9

the resulting tool-generated circuit typically has poor performance relative to what a hardware designer could achieve given the same algorithm. Furthermore, the tool-generated circuits may be inefficiently large compared to a human-designer created circuit. This results from the fact that hardware generated using prior art computer aided hardware design tools is typically based on simple line-by-line, table-driven translation and lacks the global optimizations that can be perceived by the human designer. In addition, the high-level language may lack the ability to specify non-standard sized registers, bit-serial communication paths and other design constructs known to produce smaller designs. Finally, in an attempt to approach human-designer quality results, the user of the high-level language based toolset might have to resort to a much lower-level algorithm specification, thus sacrificing the clarity of representation that is a primary benefit of this prior art toolset.

The inability of the prior art to generate efficient circuits is further illustrated by the following example. In this example, an algorithm to sum ten numbers is considered. An important aspect of the numbers to be summed in this example is that they represent student exam scores and can assume values in the range of 0 to 100 inclusive. The maximum value of the exam score total, which is specified to be the sum of ten exam scores, is thus $10*100 = 1000$. This algorithm, coded in the high-level conventional C-language, is as follows:

```
20  /* source code A */
    unsigned char exam_scores[10];
    unsigned short total_exam_scores() {
        unsigned short total = 0;
        unsigned int i;
        for (i=0;i<10;i++)
            total += exam_scores[i];
        return total;
    }
```

30 Fig. 7 shows a digital circuit 700 that may be generated by inputting the source code A into a prior art compiler. In this and subsequent circuit diagrams, a clocked register is denoted

by a rectangular box with a double line at its input and with the Δ symbol inside to denote a delay of one clock cycle. Furthermore, in circuit 700 and subsequent circuit diagrams, only the datapath logic is shown, with control logic like clocking and memory address sequencing omitted for the sake of clarity. In the source code A, an 8-bit “unsigned char” is used for the 5 “exam_scores” variable, since “unsigned char” is the smallest data type available in C that can still hold the maximum exam score of 100. A 16-bit “unsigned short” is used for the “total” variable, since this is the smallest data type that can hold the maximum exam score total of 1000. Based on this coding, a prior art compiler would generate the circuit 700 consisting of an 8-bit wide exam score memory 710, a 16-bit adder 720 and a 16-bit “total” register 730. The circuit 700, however, is far larger than optimal, because the smallest available variable sizes are larger than necessary, which results in an inefficient hardware realization.

A high-level language, i.e., PASCAL, provides the ability by way of its scalar subrange data types to specify a variable’s set of assumable values, and hence its bit-width, with greater specificity than the conventional C-language. However, PASCAL has not been implemented in an algorithm-to-hardware toolset to provide optimal circuit configurations. Because of these 15 listed shortcomings, the attempts at creating an effective tool for designing hardware using a high-level programming language have not achieved widespread use.

20 A second prior art approach to creating designs for digital hardware has been to create a specific hardware description language (HDL). Various commercially available HDLs have been developed, such as Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. After a hardware design is rendered in an HDL, the HDL design is processed into a gate-level or cell-level netlist representation using one of various hardware or manufacturer-specific synthesis 25 programs to interpret the HDL design. Then, the netlist representation is reduced to an actual hardware realization using conventional physical design tools.

HDLs and their associated computer-aided design tools have been created to be used by 30 hardware designers and require a level of expertise in the use of the particular HDL being employed to render a hardware design. Because HDLs are difficult to learn and use, only persons who frequently design hardware typically use them. If a circuit application arises in

which an HDL-representation of the algorithm is justified, the vast majority of persons must retain an expert or confront the difficult task of learning an HDL. Furthermore, these HDLs are not typically universal for the design of all target hardware technologies, since many HDLs are supported by only a single or a limited number of hardware manufacturers. Consequently, even 5 experienced users of an HDL may not be sufficiently familiar with other HDLs to be able to render a design that can be implemented in a variety of hardware systems from a number of different hardware manufacturers.

10 HDLs permit a much more detailed control over the resulting hardware implementation, but often by sacrificing the clarity and supportability of the algorithm. To further illustrate this shortcoming of conventional HDLs, the previous problem of summing 10 exam scores is reconsidered. In an attempt to create a more efficient hardware realization, the algorithm specification is modified by including a language-feature present in prior art HDLs such as 15 Verilog and VHDL, namely the ability to specify exact bit-widths of variables. Using one of a number of possible syntaxes, an integer variable whose value representation is constrained to exactly N bits is defined using the datatype “intN”. The revised algorithm for summing ten exam scores is coded in the source code B as follows:

```
20      /* source code B */
      unsigned int7 exam_scores[10];
      unsigned int10 total_exam_scores() {
          unsigned int10 total = 0;
          unsigned int i;
          for (i=0;i<10;i++)
              total += exam_scores[i];
          25      return total;
      }
```

30 The source code B uses the added bit-width language feature to specify a smaller size for the “exam_scores” and the “total” variables. In particular, a 7-bit integer type “unsigned int7” is the smallest data type available that can hold the maximum exam score of 100. Similarly, an

“unsigned int10” is the smallest data type that can hold the maximum exam score total of 1000.

Based on this coding, a prior art compiler would generate a circuit as shown in Fig. 8 consisting of a 7-bit wide exam score memory 810, a 10-bit adder 820 and a 10-bit “total” register 830.

The reduced size circuit 800 represents an improvement over the circuit 700 (Fig. 7), because the

5 circuit 800 is optimized for the data to be summed and is thus smaller. This improvement, however, comes at a price. First of all, the burden of calculating the proper bit-width of the variables has been forced on the user, and there is certainly the possibility of miscalculation. In addition, the relationship between the required bit-width of the “total” variable and the number and maximum value of the exam scores is lost. For example, assume that an exam is
10 subsequently created that has a 10-point extra credit question. A support engineer directed to update source code B might inspect the program and conclude that it requires no changes, since the 7-bit “exam_score” width is sufficient to hold the new larger maximum of 110. However, circuit 800 would fail to calculate the proper total value in all circumstances, since the new maximum 10-exam total is 1100, which exceeds the value that can be held by the 10-bit total register 830. Thus, the use of the bit-width language feature present in many conventional HDLs
15 can obscure important variable relationships in an algorithm and can lead to programming errors.

SUMMARY OF THE INVENTION

20 In one respect, the invention includes a method of converting code to a hardware realization. The method comprises steps of receiving user code including at least one algorithm specification, at least one data representation specification, and at least one data communication specification; and compiling the user code, wherein the user code is used to create a digital circuit.

25

In another respect, the invention includes another method of converting code to a hardware realization. The method comprises steps of receiving user code; identifying variables used in an operation in the user code, the operation including at least one operator; identifying a set of assumable values for each of the identified variables; calculating a set of assumable values
30 for other variables holding the results of the operation based on the identified set of assumable values; and compiling the user code, wherein the user code is used to create a digital circuit.

In yet another respect, the invention includes a system operable to create a digital circuit. The system comprises a compiler compiling user code, the user code including at least one algorithm specification, at least one data representation specification, and at least one data communication specification; and a digital circuit created based on the compiled user code.

The methods of the invention include steps that may be performed by computer-executable instructions executing on a computer-readable medium.

One embodiment of the present invention includes the independence of an algorithm description from the implementation-specific details of data representation and communication. As a hardware designer explores the implementation design space (e.g., generates various hardware realizations to produce an optimal hardware realization), the designer is not required to completely recode and obscure the algorithm to force the compiler to generate the specific structures that support a particular choice of data representation and communication.

Other certain embodiments of the present invention include the ability to precisely specify the set of assumable values for variables (i.e., the variable's assumable value set) and automatically deduce the assumable value set for some variables. Those skilled in the art will appreciate these and other aspects of various embodiments of the invention upon reading the following detailed description with reference to the below-listed drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the accompanying figures in which like numeral references refer to like elements, and wherein:

Fig. 1 illustrates an exemplary system employing principles of the invention;

10 Fig. 2 illustrates a circuit generated by a compiler of the invention;

15 Fig. 3 illustrates another circuit generated by a compiler of the invention;

20 Fig. 4 illustrates another circuit generated by a compiler of the invention;

25 Fig. 5 illustrates an exemplary method employing principles of the invention;

30 Fig. 6 illustrates another exemplary method employing principles of the invention;

35 Fig. 7 illustrates a circuit as might be generated by a prior art compiler; and

40 Fig. 8 illustrates another circuit as might be generated by a prior art compiler.

45 DETAILED DESCRIPTION OF THE INVENTION

50 In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that these specific details need not be used to practice the present invention. In other instances, well known structures, interfaces, and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

55 Fig. 1 illustrates an exemplary algorithm-to-hardware system 100 of the invention. As shown in Fig. 1, the user code 101 is entered. The user code 101 includes a high-level description that can be used to determine a configuration for the target hardware 106. The target hardware 106 may include any configurable digital circuit, such as an FPGA, or any hard-wired digital circuit, such as a custom designed VLSI chip. The target hardware 106 may be used in a target system 107, which can include processor(s), and the like. The user code 101 may include algorithm specifications 110, data representation specifications 120 and data communication specifications 130. The user code 101 uses a programming language, which may be JAVA-like, C-like, etc., but that is enhanced to support the specifications of the invention. A compiler 103

combines the specifications from the user code 101 with the circuit descriptions from the library elements called from the libraries 102 to generate a netlist 104. The system 100 may be implemented on a variety of platforms (e.g., personal computer, server, and the like) and with a variety of operating systems. The physical design tools 105 take a logical representation of a

- 5 structural design from the netlist 104 and map it to a specific hardware realization.

The algorithm specifications 110 describe variables and operators that may be included in the user code 101 for configuring the target hardware 106. The variables may be described by a set of unambiguous assumable values (i.e., an assumable value set). Operators are fully specified over the domain of their inputs and have a well-defined output range.

A variable's assumable value set, for example, may be described as a range of data values, rather than a C-language type representation (e.g., int i), where the range of represented values for "int i" is different from platform-to-platform. For example, the variable definition "int my_exam_score(0..100)" describes a variable that can assume only integer values from 0 to 100 inclusive.

The compiler 103 may also include the ability to automatically calculate an assumable value set for a variable as illustrated by the following example:

20

```
int a(0..255), b(0..255);  
int c;  
c = a + b;
```

- 25 The compiler 103 calculates that the variable "c" can assume values in the range of 0 to 510, based on the value sets specified for the "a" and "b" variables, the behavior of the "+" operator and the presence of the "c = a + b" statement.

As a result of this tightly specified or calculated value set of a variable, different register sizes or operator sizes may be realized. For example, instead of using traditional 16-bit or 32-bit registers, a 9-bit register or adder may be optimal for a particular algorithm. A variable having a

range of 2^9 values, from 0 to 511 inclusive, may be defined by the programmer in the user code 101, and a 9-bit register (e.g., for a 2's-complement data representation) may be realized through the compiler 103. A variable's assumable value set is not limited to a range of continuous numbers however. For example, a value set may be defined as consisting of the integer sequence 5 2, 4, 6 ... 20 (e.g., exemplary syntax for defining a value set having non-continuous numbers may be "int d(2,4,...,20)").

In the algorithm specifications 110, operators are defined over the domain of their inputs and have an unambiguous output range. An example of ambiguous output range, which may not be used for operators, would be the C-language sequence:

```
int i = -1;  
int j = i >> 1;
```

where the right-shift operator ">>" is left unspecified in C and may either be a signed or unsigned shift. As a result, the most significant bit of the "j" variable value is ambiguously either a "0" or a "1". Operators of the invention may include, but are not limited to, assignment, addition, subtraction, multiplication, division, and the like.

20 The data representation specifications 120 allow a user to choose how a variable is represented. For example, the variable "int a(0..255)" may be represented variously as 2's-complement, signed-digit, fully-redundant "carry-save", or the like. In an embodiment, an editor based on a graphical user interface (GUI) is provided for editing source code. For example, data representation information may be displayed in a color different from the remaining source code 25 or as part of a pop-up, dialog window. Data representation knowledge for multiple types of data representations may be stored as library elements in the libraries 102. A user may select a particular data representation, and the library element associated with the selected data representation is retrieved from the libraries 102 and used to compile the user code 101. This permits a user to leave the algorithm part of the solution untouched (i.e. the algorithm source 30 code files) and experiment with different hardware realizations through different data representations.

The data communication specifications 130 allow a user to select for each variable in the algorithm a data communication implementation. The data communication specification provides the technique for connecting hardware in the hardware realization and may include 5 conventional connection techniques (e.g., bit-serial, digit-serial, fully-parallel, and the like). For example, bit-serial includes a single data line connecting hardware elements, while fully-parallel includes multiple data lines connecting hardware elements. Library elements representing an implementation for each data communication specification may be stored in the libraries 102. The user may select a particular data communication specification through an editor (e.g., a GUI-based editor or other conventional-type editor), and the library element associated with the 10 selected data communication specification is retrieved from the libraries 102 and used to compile the user code 101. Similarly to the data representation specification 120, a user may leave the algorithm part of the source code untouched and experiment with different hardware realizations through different data communication specifications.

Because of the independence of the algorithm, data representation and data 15 communication specifications, the algorithm specifications 110 could be debugged once and left in a natural human-readable form. Any further experimentation with data representation and data communication may be performed without subsequently re-introducing bugs or clouding the 20 algorithm. This represents a significant advantage over conventional systems.

The system 100 is not limited to creating a single type of target hardware 106. The system 100 is operable to create the configuration instructions for one or more FPGAs or other types of re-configurable circuits. However, the system 100 is not limited to creating the 25 configuration instructions for re-configurable circuits, and the system 100 is additionally operable to create a chip fabricated using custom-VLSI, standard-cell or other conventional specifications, and the like.

In one embodiment of the invention, the system 100 includes the ability to automatically 30 assign a value set in the algorithm specifications 110. In the prior art example including the source code B, the programmer defined the bit-width for each of the variables. This burden on

the programmer may lead to improperly assigned bit-widths if the programmer doesn't properly readjust the bit-width for a variable affected by an algorithm change or a bit-width change of another variable. To illustrate this further, in the algorithm of source code B, assume that an exam is subsequently created that has a 10-point extra credit question. A programmer might

- 5 inspect the program and conclude that it requires no changes, since the 7-bit "exam_scores" width is sufficient to hold the new larger maximum of 110. However, the new maximum 10-exam total is 1100, which exceeds the value that can be held by the 10-bit total. Therefore, the programmer must inspect the remaining variables to insure that enough bit-width is specified.

10 The following example includes source code C and illustrates language and compiler features of the invention. This example demonstrates variables having both explicitly specified and compiler calculated value sets:

```
15      /* source code C */  
20      unsigned int(0..100) exam_scores[10];  
25      unsigned int total_exam_scores() {  
30          unsigned int total = 0;  
35          unsigned int i;  
40          for (i=0;i<10;i++)  
45              total += exam_scores[i];  
50          return total;  
55      }
```

60 The data type of the "exam_scores" variable in source code C is an array of "unsigned
65 int(0..100)". From this, the compiler understands that the value set for each exam score,
(e.g., "exam_scores[0]") is the set of integers from 0 to 100 inclusive. However, the value set for
70 the "total" variable is left unspecified in source code C. As a significant aspect of the invention,
75 the compiler 103 is able to automatically calculate that the maximum value for the "total"
80 variable is $10 * 100 = 1000$. Thus, a 10-bit adder and a 10-bit total register would be generated
85 by the invention compiler, which would support a sufficiently large maximum value of 1023.

The ability of the compiler 103 to conclude that the maximum value for the “total” variable is 1000 is based on the value set for the “exam_scores” variable. To illustrate how the compiler 103 automatically calculates the value set for the “total” variable, the loop in the source code C is “unrolled” by the compiler 103 as part of the compilation process to form the 5 equivalent internal representation:

```
10 unsigned int(0..100) exam_scores[10];
15 unsigned int total_exam_scores() {
20     unsigned int total1, total2, total3, total4, total5,
25         total6, total7, total8, total9, total10;
30     total1 = 0 + exam_score[0];
35     total2 = total1 + exam_score[1];
40     total3 = total2 + exam_score[2];
45     total4 = total3 + exam_score[3];
50     total5 = total4 + exam_score[4];
55     total6 = total5 + exam_score[5];
60     total7 = total6 + exam_score[6];
65     total8 = total7 + exam_score[7];
70     total9 = total8 + exam_score[8];
75     total10 = total9 + exam_score[9];
80     return total10;
85 }
```

The compiler 103, based on the “+” operator as set forth by this embodiment of the invention, 25 understands that “total1” has a value set of (0..100), that “total2” has a value set of (0..200), and finally that “total10” has the value set of (0..1000). Note that the “total” variable definition specifies no strict set of values in the original program. Instead, the compiler 103 calculates the value set of some variables (e.g., “total”) based on the properties of the operators that determine the variable value and the input value set of the variables feeding those operators. This allows 30 the “total10” variable (and hence the return type of the total_exam_scores() function) to be sized by the compiler to the minimum theoretical bit-width of 10-bits, without any explicit user

coding. The advantage of this is seen by considering the scenario of increasing the maximum exam score to 110. The user would indicate this by changing the exam score definition to “`unsigned int(0..110) exam_scores[10]`”. Based on this, the compiler 103 automatically calculates the new maximum exam score total to be $10 * 110 = 1100$ and sizes the “total” register and the adder that feeds it to be 11 bits wide. Therefore, an automatic and bug-free sizing of variables is facilitated.

Fig. 2 illustrates an example of a circuit 200 generated using the system 100 with source code C as the algorithm specifications 110. The circuit 200 includes a 7-bit wide exam score memory 210, a 10-bit adder 220 and a 10-bit “total” register 230. Note that the circuit 200 is identical to the circuit 800, shown in Fig. 8. Therefore, the system 100 is operable to generate a circuit as least as efficient as the prior art and with minimal error prone techniques forced on the programmer. In this example, source code C has not been explicitly augmented with any specification for the data representation or data communication. In this case, the default settings set up within the compiler (e.g., 2’s complement, fully-parallel) would come into force to supply these specifications.

In the following source code D, data representation specifications are shown:

```
20      /* source code D */  
      total_exam_scores() = carrySave;  
      total = carrySave;  
      exam_scores = twosComplement;  
      i = twosComplement;
```

25 Specifically, the “total” variable and the “total_exam_scores” function return type are
both specified to be in the fully redundant “carrySave” form. This then permits the compiler 103
to replace the carry-propagate adder 220 (Fig. 2) with a carry-save adder 320 in the resulting
implementation, shown in Fig. 3. The “exam_scores” and “i” variables are specified to be in the
30 “twosComplement” form.

Fig. 3 illustrates a circuit 300 that is an implementation of the exam score algorithm of source code C when coupled with the data representation specifications of source code D. Source code C and source code D have not been explicitly augmented with any data communication specification, so the default settings set up within the compiler (e.g., fully-parallel) would come into force to supply these specifications. The circuit 300 includes a 7-bit wide exam score memory 310, the carry-save adder 320 and 10-bit “total” registers 330 and 340. The carry-save adder 320 has no carry-chain logic, which results in a smaller input-to-output logic delay. Thus, the circuit 300 can operate at a much higher clock frequency than the circuit 200. This improved data throughput is achieved by the implementation space exploration made possible by an embodiment of the invention. Note also that an alternate data representation can be specified without modifying the algorithm source.

In the following source code E, data communication specifications are shown:

```
/* source code E */  
total_exam_scores = serial, LSDFirst, digitSize=1;  
total = serial, LSDFirst, digitSize=1;  
exam_scores = parallel;  
i = parallel;
```

Specifically, the “total” variable and the “total_exam_scores” function return type are both specified to have the data communication specification of “serial, LSDFirst, digitSize=1”. This is a more formal description of what is commonly termed “least-significant-bit-first bit-serial data communication”. This specification instructs the compiler 103 to replace the parallel carry-propagate adder 220 or the carry-save adder 320 with a single-bit-wide full adder 430 (shown in Fig. 4) in the resulting implementation.

Fig. 4 illustrates a circuit 400 as a resulting implementation of the exam score algorithm of source code C when coupled with the data communication specification of source code E. Source code C and source code E have not been explicitly augmented with any data representation specification, so the default settings set up within the compiler (e.g., 2’s

complement) would come into force to supply these specifications. The circuit 400 includes a 7-bit wide exam score memory 410, a parallel-to-serial converter 420, the carry-save adder 430, the 10-bit “total” serial shift register 450, and a 1-bit “carry out” register 440. The “total” value held in the 10-bit serial shift register 450 appears at the output serially, least-significant-bit first, over the course of 10 clock cycles. The trade-offs and advantages of a bit-serial implementation are well known to hardware designers. In general, bit-serial designs have very high clock-rates and very small combinatorial logic overhead. Throughput is often divided by some integral factor, but this degradation can sometimes be offset by multiple copies of the bit-serial circuit. Through an embodiment of the invention then, bit-serial and alternate data communication methods can be specified without modifying the algorithm source. Accordingly, different data communication implementations may be explored simply by modifying only the data communication specifications 130 of the user code 101.

It will be apparent to one of ordinary skill in the art that the system 100 encompasses each of the algorithm specifications 110, the data representation specifications 120 and the data communications specifications 130, and these specifications may not be incorporated as distinct text files or in a particular type of editor. Also, data representation and data communication specifications for every variable may not necessarily be explicitly included in user code 101, and default values can be used by the compiler 103 in such cases.

Fig. 5 illustrates an exemplary method 500 that may be facilitated using the system 100 shown in Fig. 1. In step 510, the user code 101 is received by the compiler 103. The user code 101, for example, is generated by a programmer and includes the algorithm specifications 110, the data representation specifications 120 and the data communication specifications 130. For example, the algorithm specifications 110 include variables and operators of the invention.

The data representation specifications are selected by the user and received by the compiler 103. For example, through an editor, the user selects how each variable is represented. The variables may default to one type (e.g., 2’s-complement), but the user may select other representations (e.g., signed-digit, fully-redundant carry-save, or the like). The editor may include a GUI-based editor that displays variables in different colors (e.g., one color for each

representation type), or as part of a pop-up, dialog window. Other conventional-type editors may be used. The selections are received by the compiler and used during compiling of the user code 101.

5 The data communication specifications are selected by the user and received by the compiler 103. For example, through an editor, the user selects a data communication specification for each variable. The specification may default to one type (e.g., fully-parallel), but the user may select other specifications (e.g., digit-serial, bit-serial, and the like). A GUI-based editor or other conventional-type editor may be used to select a specification. The selected specification is received by the compiler and used during compiling of the user code 101.

10
15 In step 520, the user code 101 is compiled to generate the netlist 104. This includes retrieving library elements associated with the selected data representation specifications, the selected data communication specifications and hardware elements (e.g., registers, adders, multipliers, and the like). The user code 101 is compiled with the retrieved library elements.

20
25 In step 530, the physical design tools 105 map the logical representation of the structural design provided by the netlist 104 to a form suitable for directly generating a hardware realization for the target hardware 106.

30 In step 540, the target hardware 106 is generated and may be implemented in the target system 107.

It will be apparent to one of ordinary skill in the art that the user code 101 may be edited and re-compiled. Also, the method 500 is applicable for creating digital circuits other than those embodied by configured FPGAs and including circuits realized through other re-configurable devices, one-time programmable devices, etc. Also, the method 500 is applicable for creating the specification for a VLSI chip fabricated using custom design, standard-cell design, etc.

30 Fig. 6 illustrates an exemplary method 600 for facilitating automatic value set assignment for an operator result variable in the user code 101. In step 610, variables that feed an operator

(e.g., feeding variables) for computing a result variable are identified. For example, in the source code F below, the “a” and “b” variables are feeding variables that feed the “+” operator whose result variable is “c”.

```
5      /* source code F */  
      int a(0..255), b(0..255);  
      int c, d;  
      c = a + b;  
      d = c + b;
```

In step 620, the assumable value set for each of the feeding variables is identified. For example, the value set for “a” and “b” variables is the set of integers from 0 to 255 inclusive.

In step 630, the assumable value set for the variable holding the operator result is calculated. For example, the value set for the result of the “+” operator whose result is the “c” variable is calculated to be the set of integers from 0 to 510 inclusive.

The method 600 can be used iteratively on all the operators of a calculation in an order that permits all the initially unspecified variable value sets to be calculated. In the current example of source code F, now that the value set for the “c” variable is known, the value set for the “d” variable can be calculated by following the steps of method 600 as follows.

In step 610, the “c” and “b” variables are identified as the feeding variables that feed the “+” operator whose result variable is “d”.

25 In step 620, the value set for the “b” variable is identified as being the set of integers from 0 to 255 inclusive. Based on the previous use of method 600, the value set for “c” variable is identified as being the set of integers from 0 to 510 inclusive.

30 In step 630, the assumable value set for the result of the “+” operator whose result is the “d” variable is calculated to be the set of integers from 0 to 765 inclusive.

The method 600 is not limited to sizing the variables in the code fragment shown or limited to computations with the “+” operator. It will be apparent to one of ordinary skill in the art that the method 600 may be used to size one or more variables that are the result of an

5 operation.

The methods 500 and 600 can be performed by a computer program. The computer program can exist in a variety of forms both active and inactive. For example, the computer program and objects can exist as software comprised of program instructions or statements in source code, object code, executable code or other formats; firmware program(s); or hardware description language (HDL) files. Any of the above can be embodied on a computer readable medium, which include storage devices and signals, in compressed or uncompressed form. Exemplary computer readable storage devices include conventional computer system RAM (random access memory), ROM (read only memory), EPROM (erasable, programmable ROM), EEPROM (electrically erasable, programmable ROM), and magnetic or optical disks or tapes. Exemplary computer readable signals, whether modulated using a carrier or not, are signals that a computer system hosting or running the computer program can be configured to access, including signals downloaded through the Internet or other networks. Concrete examples of the foregoing include distribution of executable software program(s) of the computer program on a CD ROM or via Internet download. In a sense, the Internet itself, as an abstract entity, is a computer readable medium. The same is true of computer networks in general.

While this invention has been described in conjunction with the specific embodiments thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. There are changes that may be made without departing from the spirit and scope of the invention.